AN IMPROVED MAXIMAL MATCHING PARALLEL ALGORITHM

by

A. Israeli and Y. Shiloach

Technical Report #333

September 1984

* Computer Science Department, Technion-IIT, Haifa 32000, Israel
** Elbit Computer Ltd., P.O.B. 5390, Haifa 31053, Israel.
AN IMPROVED

MAXIMAL MATCHING PARALLEL ALGORITHM

by

A Israeli
Computer Science Department
Technion, Haifa 32000, Israel

Y. Shiloach
Elbit Computers Ltd.
P.O.B. 5390, Haifa 31053, Israel

ABSTRACT

A parallel $O(\log^3 |E|)$ algorithm for finding a maximal matching in a graph $G(V,E)$ is presented. The model of computation is the CRCW-PRAM and $|V|+|E|$ processors are used.

This algorithm is a substantial improvement upon the two previous algorithms known to us. These algorithms, [KW-84] and [Lev-80], achieve depth of $O(\log^4 |E|)$ with $\frac{|E|^3}{\log |E|}$ and $|E|+|V|$ processors respectively. The last one though having a better performance than the first, applies only to bipartite graphs.
1. INTRODUCTION

A matching of an undirected graph \( G(V, E) \) is a set of edges \( M \subseteq E \) in which no two edges have a common vertex. A matching \( M \) of a graph \( G(V, E) \) is maximal if it is not strictly contained in any other matching.

Finding a maximal matching sequentially is done simply by inspecting the edges of the graph one by one. An inspected edge is added to the matching if it is not adjacent to any of the edges that are already there. Unfortunately, a straightforward parallel implementation of this algorithm can hardly be seen.

A parallel algorithm for maximal matching in bipartite graphs is given in [Lev-80]. This algorithm has depth of \( \log^4 |E| \) using \( |V| + |E| \) processors. A solution for general graphs can be obtained from [KW-84] by a relatively simple construction. The depth of this solution is also \( \log^4 |E| \) but as many as \( \frac{|E|^3}{\log |E|} \) processors are needed.

Our algorithm is a substantial improvement as it applies to general graphs; its depth is \( \log^3 |E| \) with \( |V| + |E| \) processors. It uses a new technique for finding Euler circuits in graphs, presented in [AIS-84], as a subroutine. The model of computation inherited from [AIS-84], is that of shared memory in which simultaneous REED/WRITE from/to the same memory location are allowed, (CRCW-PRAM). In the latter case we do not care which processor actually writes.

The algorithms of [Lev-80] and [KW-84] are implemented in a weaker model, namely EREW-PRAM. Implementing our algorithm in this model multiplies the time by another logarithmic factor. Yet it remains a significant improvement to both algorithms. Moreover it is not clear how these algorithms can benefit from a CRCW-PRAM.

2. AN OUTLINE OF THE ALGORITHM

The maximal matching algorithm is divided into phases. Each phase, \( \phi_i \), starts with an initial graph \( G_i \) and finds a (not necessarily maximal) matching \( M_i \) of it. The edges of \( M_i \) are removed from \( G_i \) together with all their incident edges. The
matching \( M_i \) is accumulated as a subset of the final maximal matching of \( G \). The graph obtained from \( G_i \) by this removal of edges is \( G_{i+1} \) which is the initial graph of the next phase namely \( \Phi_{i+1} \). The algorithm terminates when all edges are removed.

The core of each phase is the so called \textit{DEGREE}-\textit{SPLIT} routine.

Let \( G \) be a graph of maximal degree \( \Delta(G) \) satisfying:

\[ 2^k \leq \Delta(G) \leq 2^{k+1} + 1 \quad \text{for some integer } k > 0 \]

When \textit{DEGREE}-\textit{SPLIT} is applied to \( G \) it removes some of its edges (but none of its vertices) yielding a subgraph of \( G \), called \( G^1 \) for which:

\[ 2^{k-1} \leq \Delta(G^1) \leq 2^k + 1 \]

Each phase \( \Phi_i \) consists of at most \( 1 + \lceil \log \Delta(G) \rceil \) successive applications of \textit{DEGREE}-\textit{SPLIT}. The output graph of each application is the input graph to the next. The first input graph is \( G_i \) and the output of the last application of \textit{DEGREE}-\textit{SPLIT} is the matching \( M_i \).

The following definition is needed for the outline of \textit{DEGREE}-\textit{SPLIT}:

Let \( k \) be the minimal integer satisfying \( 2^k \leq \Delta(G) \leq 2^{k+1} + 1 \) for a given graph \( G \).

Then all the vertices for which \( d(u) \geq 2^k \) are called \textit{active vertices}.

\textit{DEGREE}-\textit{SPLIT} can be outlined as follows:

1. Find all the active vertices of \( G \).
2. Construct the graph \( G_a \) induced by all edges of \( G \), incident with at least one active vertex.
3. Make \( G_a \) a graph of even degrees by connecting all the odd-degree vertices to a dummy vertex \( v^* \).
4. Find an Euler circuit in every connected component of \( G_a \).
5. In each connected component label the edges of \( G_a \) by 0 and 1 alternately following the route determined by its Euler circuit.
In the component containing $v^*$, start your tour from it, labeling the first edge by 0. In the other components start from any active vertex and label the first edge by 1.

(6) Remove all the 0-labeled edges from $G$.

3. THE COMPLEXITY OF THE ALGORITHM

The $O(\log^3 |E|)$ depth of the algorithm follows directly from theorems 1, 2 (Corollary 2.1-d) and 5 below.

Theorem 1:

DEGREE-SPLIT can be implemented on a CREW-PRAM with $|V| + |E|$ processors in $\log |E|$ depth.

Proof:

The logarithmic depth of step 4 relies on the result of [AlS-84]. All the other steps can easily be implemented in logarithmic time, assuming that the input graph is given in a form of neighbours lists for each vertex.

Theorem 2:

Let $G$ be a graph such that $2^k \leq \Delta(G) \leq 2^k + 1$, $k \geq 2$. Let $G^j$ be the output graph of DEGREE-SPLIT$(G)$. If $v$ is an active vertex of $G$ then:

$$2^k - 1 \leq d_{G^j}(v) \leq 2^k + 1 \quad (\ast)$$

Corollary 2.1:

Let $G^j$ denote the graph obtained from $G$ by $j$ successive applications of DEGREE-SPLIT.

a. The graph $G^j$ satisfies $2^k - 1 \leq \Delta(G^j) \leq 2^k + 1$ and therefore all the active vertices in $G$ remain so in $G^j$. Moreover, they are active in $G^j$ for all $1 \leq j \leq k$. Thus,
b. Every active vertex $v$ in $G^k$ satisfies: $1 \leq d_G(v) \leq 3$. In particular: $\Delta(G^k) \leq 3$.

c. Since $\Delta(G^k) \leq 3$, each vertex of $G^k$ with a positive degree is active in $G^k$.

d. At most two applications of $DEGREE\text{-}SPLIT$ on $G^k$ are necessary in order to produce a matching.

Proof of Theorem 2:

Let $v \in V$ be an active vertex of $G$. Note that $d_G(v) \leq d_{G'}(v) \leq d_G(v)+1$. Consider the following cases:

Case 1: $v$ is in the same component as $v'$.

Case 1.1: $v$ is not adjacent to $v'$.

In this case $d_G(v)$ is even. The edges incident with $v$ can be divided to pairs of the form $(u,v)$ $(v,w)$ where $(u,v)$ precedes $(v,w)$ on the corresponding Euler circuit. Exactly one edge of each such pair is labeled 0 so we get $d_{G'}(v) = \frac{1}{2} d_G(v)$ which obviously satisfies ($*$).

Case 1.2: $v$ is adjacent to $v'$.

In this case the edges incident with $v$ can be partitioned as in Case 1.1 to pairs but one edge is a dummy edge. Thus the resulting number of 0-labeled edges is either bigger or smaller by 1 than the number of 1-labeled edges. Since $d_G(v)$ is odd we get either $d_{G'}(v) = \left\lfloor d_G(v)/2 \right\rfloor$ or $d_{G'}(v) = \left\lfloor d_G(v)/2 \right\rfloor + 1$. In any case ($*$) is satisfied.

Case 2: $v$ is not in the same component as $v'$.

Case 2.1: The 0-1 labeling does not start at $v$.

This case is identical to Case 1.1.

Case 2.2: The 0-1 labeling starts at $v$.
In case that \( v \)'s component has an even number of edges, it is like Case 1.1. In case that \( v \)'s component has an odd number of edges we get \( d_{G_k}(v) = \frac{1}{2} d_G(v) + 1 \). In both cases (*) is satisfied. 

Let \( A_k \) be the set of vertices that become active during \( \Phi_k \). In view of Corollary 2.1-a and 2.1-c \( A_k \) is exactly the set of all the vertices that are active in \( G^k \).

A set \( S \subseteq V \) is a vertex cover of a graph \( G(V,E) \) if for every edge \( (u,v) \in E \), either \( u \in S \) or \( v \in S \).

**Theorem 3:**

The set \( A_k \) is a vertex cover of \( G_k \).

**Proof:**

Assume to the contrary that there exists an edge \( (u,v) \) such that \( u \) and \( v \) are never active during \( \Phi_k \). Thus the edge \( (u,v) \) is in \( G^j - G_k \) for all \( 1 \leq j \leq k \), and is not deleted by any application of DEGREE-SPLIT. In particular, \( d_{G_k}(u) \geq 1 \) and \( d_{G_k}(v) \geq 1 \). By Corollary 2.1-c both \( u \) and \( v \) are active in \( G^k \) - a contradiction.

Let \( H(V,E') \) be a subgraph of \( G(V,E) \). For any set \( S, S \subseteq V \), we denote the set \( S \cap V \) by \( S \setminus H \). (The restriction of \( S \) to \( H \)).

**Theorem 4:**

A slight modification of the last two applications of DEGREE-SPLIT on \( G^k \) can assure that at least half of the vertices of \( A_k \) are incident with edges of \( M_k \). In other words:

\[
|A_k \setminus G_{k+1}| \leq \frac{1}{2} |A_k| \quad (**) 
\]
Proof:

If (***) holds for each connected component of \( G^k \), it holds for the entire graph. Hence it can be assumed that \( G^k = (V^k, E^k) \) is connected and by Corollary 2.1-c \( V^k = A_k \). The matching \( M_k \) is produced by two additional applications of DEGREE-SPLIT on \( G^k \). If on each application, the smaller of the two sets of 0-labeled and 1-labeled edges is removed rather than the always 0-labeled, then \( |M_k| \geq 1/4|E^k| \). Thus, if the number of edges in \( G^k \) is at least the number of vertices \( (|A_k|) \) then at least \( 1/4|A_k| \) edges belong to \( M_k \) and they are incident with at least \( 1/2|A_k| \) vertices. Otherwise \( |E^k| = |A_k| - 1 \) and \( G^k \) is a tree. In this case choose any edge incident with a leaf and put it in \( M_k \). Since \( \Delta(G^k) \leq 3 \), the removal of this edge from the tree breaks it into at most two trees, say \( T_1 \) and \( T_2 \), having \( t_1 \) and \( t_2 \) vertices, respectively. \((t_1 + t_2 = |A_k| - 2)\). Applying the modified DEGREE-SPLIT algorithm on each tree, results in at least \( 1/4(t_1 - 1) \) edges of \( T_1 \), \((1/4(t_2 - 1) \) edges of \( T_2 \) in \( M_k \). These edges are incident with \( 1/2(t_1 + t_2 - 2) \) vertices altogether. So, together with the two vertices of the first edge one has at least \( 1/2|A_k| \) vertices in \( M_k \).

Theorem 5:

There are at most \( \log_2 |V| \) phases in the whole algorithm.

Proof:

It is enough to show that for all \( i \), \( 1 \leq i \leq \log_2 |V| \) the graph \( G_i \) has a vertex cover \( C_i \) such that \( |C_i| \leq (2/3)^{i-1} |V| \). The proof proceeds by induction on \( i \). For \( i=1 \) we take \( C_1 = V \). Let us assume that the vertex covers \( C_1, ..., C_i \) for \( G_1, ..., G_i \) are already defined. The idea of the proof is to use either \( G_i / G_{i+1} \) or \( A_i / G_{i+1} \) as a vertex cover for \( G_{i+1} \). The decision which of them to choose depends on the relative sizes of \( A_i \) and \( C_i \). Theorem 4 is used to assure that in any case, a new small enough vertex cover can always be chosen. More formally, we have to show that \( G_{i+1} \), the initial graph after \( \Phi_{i+1} \), has a vertex cover \( C_{i+1} \) that satisfies:
Consider the following cases:

**Case 1:** $|A_k| \leq 4/3 |G_k|$

By Theorem 4 at least half of the vertices in $A_k$ are removed by the end of $T_k$. The set $A_k \setminus G_{k+1}$ is a vertex cover for $G_{k+1}$ and it satisfies:

$$|A_k \setminus G_{k+1}| \leq \frac{1}{2} |A_k| \leq \frac{1}{2} \cdot \frac{4}{3} |G_k| = \frac{2}{3} |G_k|$$

**Case 2:** $|A_k| > 4/3 |G_k|$

By Theorem 4 at least half of the vertices in $A_k$ are removed by the end of $T_k$. These vertices are all the vertices incident with edges of the matching $M_k$ of $G_k$. The set $G_k$ is a vertex cover of $G_k$. If $(u, v) \in M_k$ and $u \notin G_k$ then $v \in G_k$. Hence at least half of the vertices of $M_k$ (i.e. at least $1/4 |A_k|$) are in $G_k$. The set $G_k \setminus G_{k+1}$ is a vertex cover of $G_{k+1}$ and it satisfies:

$$|G_k \setminus G_{k+1}| \leq |G_k| - \frac{1}{4} |A_k| \leq |G_k| - \frac{1}{4} \cdot \frac{4}{3} |G_k| = |G_k| - \frac{1}{3} |G_k| = \frac{2}{3} |G_k|$$

**ACKNOWLEDGEMENT**

We would like to thank Dr. Gabby Lev for helpful discussions and Dr. Manfred Warmuth for his careful reeding of the manuscript.

**REFERENCES**

